

A parallel time series algorithm for searching similar sub-sequences

Firas Mahmood Saeed¹, Salwa M. Ali^{2,3}, Mohammed W. Al-Neama⁴

¹Medicine College, Mosul University, Mosul, Iraq

²Department of Computer Science, Unaizah College of Sciences and Arts, Qassim University, Qassim, Saudi Arabia

³Faculty of Science, Ain Shams University, Cairo, Egypt

⁴Education College for Girls, Mosul University, Mosul, Iraq

Article Info

Article history:

Received Mar 30, 2021

Revised Dec 16, 2021

Accepted Jan 7, 2022

Keywords:

Dynamic time warping
Message passing interface
OpenMP
Parallel algorithm
Similarity search
Time series

ABSTRACT

Dynamic time warping (DTW) is an important metric for measuring similarity for most time series applications. The computations of DTW cost too much especially with the gigantic of sequence databases and lead to an urgent need for accelerating these computations. However, the multi-core cluster systems, which are available now, with their scalability and performance/cost ratio, meet the need for more powerful and efficient performance. This paper proposes a highly efficient parallel vectorized algorithm with high performance for computing DTW, addressed to multi-core clusters using the Intel quad-core Xeon co-processors. It deduces an efficient architecture. Implementations employ the potential of both message passing interface (MPI) and OpenMP libraries. The implementation is based on the OpenMP parallel programming technology and offloads execution mode, where part of the code sub-sequences on the processor side, which are uploaded to the co-processor for the DTW computations. The results of experiments confirm the effectiveness of the algorithm.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Mohammed W. Al-Neama
Education College for Girls, Mosul University
Mosul, Iraq
Email: mweama@uomosul.edu.iq

1. INTRODUCTION

The search for similar subsequences is required in many applications of the intellectual analysis of time series, such as financial forecasts [1], medical research [2], climate modeling [3], as well as the robustness of dynamic time warping (DTW) to some local time-series filtering [4]. The task of finding a similar subsequence can be informally defined as: Let there be a time series and a search query. It is necessary to find a subsequence of the source series, which is as similar as possible in the form of a search query.

DTW measure [5] is recognized by the scientific community as the best measure of time series similarity for many different areas [6] since it makes it possible to adequately measure the similarity of time series that have different lengths or shifts, stretching, or compression relative to each other. Since the DTW measure has quadratic computational complexity concerning the length of the search query, methods for reducing the computational load were proposed such as indexing [7], early rejection of redundant calculations, and reuse of the results of previously performed calculations [8]. The sequential DTW algorithm [9], integrating many of these techniques, is the fastest sequential search algorithm for similar subsequences [10].

From another point of view, the swift growth of the hardware of the high-performance computing (HPC) supplies high-performance cost proportion computational ability, such as, the efficient pipelined implementations of algorithms in [4] and the efficacious program to compute similarity matrix using multi-

core system [11]. As well as, parallel algorithms were proposed for finding similar subsequences for field-programmable gate array (FPGA) [12], multi-core of the graphics processing unit (GPU) architectures [13], and Intel many integrated core (MIC) [14]. The MIC discussed a competitive alternative to the more common NVIDIA GPU and FPGA systems. Intel MIC systems are based on the common Intel X86 architecture and support related models and parallel programming tools. Intel offers two generations of MIC devices with the Xeon base name: the knights corner (KNC) [15] coprocessor with 57-61 cores and the knights landing (KNL) standalone processor system [16] with 64-72 cores, respectively. MIC systems, as a rule, give the highest performance in applications where there is a large amount of data involved in the calculations that can be vectorized by the compiler [17]. Vectorization refers to the ability of the compiler to replace several scalar operations in the body of the cycle with a fixed number of repetitions in one vector operation [18]. The work in [19] presents parallel search algorithms for similar subsequences in ultra-large time series on cluster systems.

Shabib *et al.* [20] a method for parallelizing a sequential search algorithm for similar subsequences of the UCR-DTW software for a computing cluster based on the use of the spark framework. The algorithm assumes sub-sequence of the time series, however, the number of subsequences does not coincide with the number of computational nodes of the cluster system. The experiments show enhance algorithm is 2-5 times ahead of UCR-DTW.

In the study Zymbler [21] a parallel algorithm was developed to search for similar subsequences on a computing cluster with nodes based on the extension called KNC coprocessors. The proposed computational schedule "CPU+Phi" assumes sub-sequence of the series by cluster nodes and the following separation of functions between the processor and KNC inside the node. KNC calculates the DTW measure for subsequences loaded into its memory by the processor, and selects a subsequence as close as possible to the search pattern. In [22], the authors proposed an approach to the efficient search for similar subsequences in the ram of the KNL processor.

This paper proposes a new parallel algorithm for finding similar subsequences in ultra-large time series on cluster systems with nodes based on Intel Xeon multi-core knights landing processors, and called (B_Match). Computations are parallelized at two levels: at the level of all cluster nodes-using message passing interface (MPI) technology, within a single cluster node-using OpenMP technology. The algorithm assumes the use of additional data structures and redundant calculations, which make it possible to efficiently use the capabilities of vectorization of calculations on KNL processor systems.

The paper is organized as shown in: Section 2 provides a formal statement of the problem and gives a brief description of the sequential UCR-DTW algorithm used as the basis for a new parallel algorithm. Section 3 describes the B_Match algorithm. Section 4 describes computational experiments investigating the effectiveness of the proposed algorithm. The conclusion summarizes the results obtained in the study.

2. STATEMENT OF THE PROBLEM

2.1. Formal definitions and notations

Let us define the terms used by [9]. The time series T is a chronologically ordered sequence of real values: $T = t_1, t_2, \dots, t_n$; $t_i \in R$. The number m is denoted $|T|$ and is called the length of the time series. A similarity measure S between two-time series A and B is a real non-negative function that calculates the distance between these series: $S(a, b) \geq 0$, where $a \in A$ and $b \in B$.

Let two-time series be given $A = (a_1, a_2, \dots, a_n)$ and $B = (b_1, b_2, \dots, b_n)$. Then the DTW is the measure of similarity DTW (A, B), calculated in [5]: $DTW(A, B) = d(n, n)$,

$$\begin{aligned} d(x, y) &= (a_x - b_y)^2 + \min\{d(x-1, y), d(x, y-1), d(x-1, y-1)\}; \\ d(0, 0) &= 0; d(x, 0) = d(0, y) = \infty; 1 \leq x, y \leq n. \end{aligned} \quad (1)$$

as shown in (1), the matrix $(d_{xy}) \in R^{n \times n}$ expresses the correspondence between the points of the compared time series and is called a transformation matrix. The warping path P is a sequence of elements of the transformation matrix that determines the correspondence between the time series Q and C . Let the element p_t of the transformation path p be defined as $p_t = (x, y)_t$, then (2).

$$P = p_1, p_2, \dots, p_T \dots, p_L; n \leq T \leq 2m - 1 \quad (2)$$

The amount of computation in calculating the measure of similarity DTW can be reduced due to the coarsening of the similarity. For this, additional restrictions may be imposed on the transformation path. One of the most commonly used restrictions is the so-called sacochiba band [23], which does not allow the transformation path to deviate by more than r elements from the diagonal of the transformation matrix.

A subsequence $T_{x,m}$ of a time series T is a continuous subset T of length m starting at position x : $T_{x,m} = (t_x, t_{x+1}, \dots, t_{x+m-1})$, $1 \leq x \leq n - m + 1$, $m \ll n$. Let there be a long time series T and a user-defined substantially shorter time series Q (called a search query), where $n = |T| \gg |Q| = m$. Then the best match (similar subsequence) is the subsequence $T_{x,m}$, the form of which is as similar as possible to the query Q in the sense of the measure DTW:

$$\exists x \forall i \text{ DTW}(Q, T_{x,m}) \leq \text{DTW}(Q, T_{i,m}) \text{ where } 1 \leq x, i \leq n - m + 1 \quad (3)$$

further, the subsequence $T_{x,m}$ will be called a candidate (for max similarity to the query Q) and denoted by C .

2.2. Word-processing software

The UCR-DTW algorithm Silva *et al.* [9] is one of the fastest algorithms for searching for similar subsequences, integrating many techniques that accelerate the search. Its main ideas are briefly described below. Using the Euclidean distance squared. Euclidean distance (ED) between two-time series Q and C , where $|Q| = |C|$ is defined as (4).

$$ED(Q, C) = \sqrt{\sum_{x=1}^m (q_x - c_x)^2} \quad (4)$$

The square root calculation in ED (4) and DTW (1) can be deleted since this will not change the relative ranking of subsequences similar to the query since both functions are monotonic and concave. The query and subsequence of the time series must be subjected to Z-normalization [24]. Z-normalization of the time series T is called the time series $\hat{T} = (\hat{t}_1, \hat{t}_2, \dots, \hat{t}_n)$, whose elements are calculated in (5).

$$\hat{t}_x = \frac{t_x - \mu}{\sigma}; \text{ where } 1 \leq x \leq n; \mu = \frac{1}{n} \sum_{x=1}^n t_x; \sigma^2 = \frac{1}{n} \sum_{x=1}^n t_x^2 - \mu^2. \quad (5)$$

Z-normalization allows you to compare the shapes of the series, which are different in amplitude. After normalization, the arithmetic mean of the time series is approximately 0, and the standard deviation is close to 1. The lower bound (LB) of similarity is a function whose computational complexity is less than the computational complexity of the DTW measure, the LB is used to drop candidates that are obviously unlike the query without calculating the DTW measure [6].

We denote by B_{CLB} the best current lower bound for the similarity of the current subsequence $T_{x,m}$ and the search query Q . If the lower bound for the candidate exceeds the threshold B_{CLB} , then the value of the DTW measure for this candidate will also exceed B_{CLB} . during the time series scan, the UCR-DTW algorithm tries to improve (decrease) the B_{CLB} value. The B_{CLB} value is initialized as $+\infty$ and is calculated as shown in the x^{th} search step.

$$B_{CLB_x} = \min \left(B_{CLB_{x-1}}, \begin{cases} +\infty & \text{if } LB(Q, T_{x,m}) > B_{CLB_{x-1}} \\ \text{DTW}(Q, T_{x,m}) & \text{otherwise} \end{cases} \right) \quad (6)$$

The UCR-DTW algorithm uses the following lower bounds for similarity: $LB_{kim}FL$ [9], $LB_{keogh}EC$, and $LB_{keogh}EQ$ [7], which are applied in cascading fashion. The lower boundary of $LB_{kim}FL$ is the Euclidean distance between the first and last pairs of points Q and C .

$$LB_{kim}FL(Q, C) = ED(\hat{q}_1 - \hat{c}_1) + ED(\hat{q}_m - \hat{c}_m) \quad (7)$$

The lower bound of $LB_{keogh}EC$ shows the minimum similarity between the envelope of the request E (Envelope) and the candidate \hat{C} and is calculated in (8).

$$LB_{keogh}EC(Q, C) = \sum_{x=1}^m \begin{cases} (\hat{c}_x - u_x)^2, & \text{if } \hat{c}_x > U_i \\ (\hat{c}_x - l_x)^2, & \text{if } \hat{c}_x < L_i \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

In (8), the sequences $U = u_1, \dots, u_m$ and $L = l_1, \dots, l_m$ denote the upper and lower boundaries of the query shell Q , which are calculated by (9).

$$U_x = \max(\hat{q}_{x-r}, \dots, \hat{q}_{x+r}); L_x = \min(\hat{q}_{x-r}, \dots, \hat{q}_{x+r}) \quad (9)$$

The lower boundary of $LB_{Keogh}EQ$ represents the Euclidean distance between the Q query and the candidate C shell, i.e., in comparison with $LB_{Keogh}EC$, the roles of the request and the candidate are reversed.

$$LB_{Keogh}EQ(Q, C) := LB_{Keogh}EC(C, Q) \quad (10)$$

3. PARALLEL ALGORITHM B_MATCH

This section presents a new parallel algorithm for finding the most similar subsequence of the time series for cluster systems with nodes based on multi-core systems of the Intel MIC architecture. Parallelization of the algorithm is based on parallelism over data, data alignment in memory, and vectorization of computations. To reduce the amount of computation, distribute the subsequence into available nodes by MPI technology [25] each node runs simultaneously to find the similarity. Concurrency according to the data inside the node of the cluster system is implemented as threads of the node are run on the cores of the node by shared memory of the data by using OpenMP [26]. The efficiency of vectorization of cycles can be significantly reduced due to unaligned access to data in RAM, which gives rise to the effect of loop separation [18].

3.1. Time series decomposition

The decomposition of the time series provides parallelism according to the data at the level of the computing nodes of the cluster system. To prevent the loss of the resulting subsequences located at the junction of subsequences, the technique of splitting with overlap is proposed, which is: at the end of each subsequence of the time series, except the last one in order, $n - 1$ elements of the series are taken from the beginning of the next subsequence, where n is the length of the query. The formal definition of a split with overlap is as: let $N = |T| - m + 1 = n - m + 1$ is the number of subsequences that need to be processed, F is the number of subsequences, $T^{(i)}$ is the i^{th} subsequence of $T = (t_1, t_2, \dots, t_m)$; $0 \leq i \leq F - 1$, then the *start* and *length* numbers of the subsequence $T^{(i)}$ is defined by (11).

$$start\ no. = i \cdot \lfloor N/F \rfloor + 1; \text{ leng.no.} = \begin{cases} \lfloor \frac{N}{F} \rfloor + (N \bmod F) + n - 1, & \text{if } i = F - 1 \\ \lfloor \frac{N}{F} \rfloor + n - 1 & \text{if } 1 \leq i < F - 1 \end{cases} \quad (11)$$

The number of subsequences (computational nodes of the cluster system) F is selected so that the subsequence and the corresponding auxiliary data of the algorithm can be placed in the RAM of the computational node.

3.2. Data composition

The proposed data arrangement in the RAM of the computational node of the cluster system provides a representation of a subsequence of the time series and auxiliary data of the algorithm in the form of memory-aligned matrices, the processing cycles of which are vectorized by the compiler. To perform the data alignment, let the subsequence $T_{x,m}$ of the series T be processed using a vector register containing w real numbers. if the length of the subsequence is not a multiple of ω , then the subsequence is complemented by dummy zero elements. We denote the number of dummy elements by $pad = \omega - (n \bmod \omega)$, then the aligned subsequence $\hat{T}_{x,m}$ is defined by (12):

$$\hat{T}_{x,m} = \begin{cases} t_x, t_{x+1}, \dots, t_{x+m-1}, \underbrace{0, 0, \dots, 0}_{pad}, & \text{if } m \bmod \omega > 0; \\ t_x, t_{x+1}, \dots, t_{x+m-1}, & \text{otherwise.} \end{cases} \quad (12)$$

by definition (1), $\forall Q, C: DTW(Q, C) = DTW(\hat{Q}, \hat{C})$. We assume that the query and candidate subsequences are aligned in memory, and for simplicity, we use the notation Q and C , respectively. The alignment of subsequences allows us to avoid the overhead of breaking cycles into several iterations. All aligned subsequences of the time series are stored as a matrix of subsequences to provide vectorization of the calculations. The subsequence matrix $S_T^m \in R^{N \times (m+pad)}$ is defined as (13).

$$S_T^m(x, y) := \hat{t}_{x+y-1}. \quad (13)$$

The number of lower boundaries of similarity used in the search algorithm for the most similar subsequence, for $lb_{max}(lb_{max} \geq l)$, and for $LB_1, LB_2, \dots, LB_{lb_{max}}$, we denote these boundaries listed in the order

of their calculation in the cascade of application of lower bounds. Then $L_T^n \in R^{N \times lb_{max}}$, the matrix of lower bounds for the similarity of all subsequences of length n of the time series T with the search query Q , as shown in (14).

$$L_T^n(x, y) := LB_y(T_{x,n}, Q). \quad (14)$$

The similarity map is a column vector $B_T^n \in B^N$, which for each subsequence of n of the T stores the conjunction of applying all the lower boundaries of the similarity of this subsequence to the current (B_{CLB}):

$$B_T^n(x) := \bigwedge_{y=1}^{lb_{max}} (L_T^n(x, y) < bsf). \quad (15)$$

we introduce a matrix for storing candidate subsequences that is those subsequences from the matrix S_T^n that were not discarded when applying the lower boundary of similarity. This matrix will be processed in parallel to calculate the values of the DTW measure between the candidates and the query. Furthermore, the minimum of the calculated values of the DTW similarity measure is used as the B_{CLB} threshold.

Parallel processing of the matrix of candidates uses a breakdown of the rows of this matrix into sections processed by individual threads of one node of the cluster. We denote the number of threads used by the parallel algorithm by p ($p \geq 1$). We introduce the size of the segment $s \in N$ ($s \leq \lfloor N/p \rfloor$) - the number of rows of the candidate matrix processed by one thread. Then the candidate matrix $C_T^n \in R^{(s,p) \times (n+pad)}$ is defined as (16):

$$C_T^n(x, .) := S_T^n(k, .) : B_T^n(x) = TRUE. \quad (16)$$

although the calculation of the candidate matrix is a parallelizable operation, the vectorization of the implementation of the corresponding cycle is difficult, because, according to definition (1), the data dependency takes place when calculating the similarity measure DTW.

3.3. The implementation

The parallel implementation of the search for similar subsequences of the time series proposed in this paper is presented in Algorithm 1. To parallelize the operations described above, the matrix of lower boundaries of similarity is divided into segments. By construction, the matrix of lower boundaries of similarity has a significantly larger number of rows than the matrix of candidates. Accordingly, after filling and calculating the candidate matrix, and updating the threshold B_{CLB} , each thread should continue scanning its segment until it is exhausted. to store the number of the last processed candidate in the segment, an index array $Pos \in N^p$ is introduced, where:

$$pos_x := k : p \cdot (x - 1) + 1 \leq k \leq \left\lfloor \frac{N}{x \cdot p} \right\rfloor \wedge \forall y, 1 \leq y \leq lb_{max}, LB_T^n(k, y) < bsf. \quad (17)$$

the absence of candidates means that the processing of the subsequence is completed. Otherwise, the DTW similarity measure is calculated for each row of the candidate matrix. To display the index of the most similar subsequence query, the index array $Idx \in N^{s \cdot p}$, is introduced, which is designed to store the position of the subsequence in the time series and is defined as (18).

$$idx_x := k : 1 \leq k \leq N \wedge \exists S_T^n(x, .) \Leftrightarrow \exists T_{x,n} \Leftrightarrow k = (x - 1) \cdot n + 1. \quad (18)$$

After the candidate matrix is filled, for each of its rows the value of the measure of similarity DTW between the request and the candidate is calculated. Loop parallelization is performed using the OpenMP `#pragma omp parallel for` directive, which provides static splitting of loop iterations between threads. If the calculated similarity value is less than the current B_{CLB} value, then B_{CLB} is replaced with the calculated value.

The best among all subsequences of the series of lower similarity ratings and the corresponding subsequence is selected using the MPI_Allreduce reduction operation, which returns the minimum value of the threshold B_{CLB} among all computational processes and the corresponding subsequence, copying them to the memory of each process. The fact that the processing of a series has been completed is also determined using the global reduction operation, in which a conjunction of flags to complete processing by each process of its subsequence of the series is performed.

Algorithm 1. B_Match

```

Input  : Time series ( $T$ ), Query ( $Q$ ), segments,  $F$ 
Output :  $B_{CLB}$ ,  $B_{match}$ 
1: initialize  $i \leftarrow 0$ 
2:  $N \leftarrow |T^{(i)}| - m + 1$ ,  $sub\_seq \leftarrow T_{(1..N),m}^{(i)}$  and  $B_{CLB} \leftarrow DTW(sub\_seq, Q)$ 
3:  $n \leftarrow N$ 
4: While  $i \leq F$  do in parallel ▷ Distributing  $T^{(i)}$  over  $M$  nodes
5:   Get ( $T^{(i)}$ )
6:    $i = i + 1$ 
7:   While candidate = empty do in parallel ▷ Distributing distance computation over  $P$  cores
8:     Process Candidate
9:     Compute DTW (candidate) by using (1) and Update Distances
10:    DTW [ $T^{(i)}$ ] = DTW (candidate)
11:   end While
12:   Compute  $\{B_{CLB}, B_{match}\}$  by using (6)–(10)
13: end While
14: return  $\{B_{CLB}, B_{match}\}$ 

```

4. COMPUTATIONAL EXPERIMENTS

The proposed algorithm is implemented in C++ by using OpenMP and MPI libraries. The proposed paradigm matches the features of the system of the multi-core cluster very well. The performing program has the utilization of the parallelism of coarse-grained on a process level, in which each MPI procedure is implemented on one multi-core. Also, it has the utilization of the parallelism of fine-grained on a "while level", in which each MPI process produces a set of threads to overrun the multicore processors when using parallel code by OpenMP. Furthermore, should be analyzed the obtained results carefully to understand the items that interfere with the determined comparison model.

4.1. Hardware platform

The code of the introduced algorithm has been run on Sun Microsystems cluster, at Bibliotheca Alexandrina, Egypt, which is provided by LinkSCEEM-2 systems. This platform has 130 nodes and 64 GB memory (the allowed number of nodes for one user is 32 nodes): each node contains two Intel quad-core Xeon 2.83 GHz processors (64-bit technology), with 8 Gbyte RAM, 80 Gbyte hard disk, and a dual-port in fin band (10 Gbps), and the operating system is 64-bit Linux انقر أو اضغط هنا لإدخال نص.

In the experiments, the efficiency of the B_Match algorithm was studied both on one computing node of the cluster system and on the cluster system as a whole. A study on one computing node of a cluster system allows you to determine how effectively parallel computing performed by the Intel Xeon multi-core processor system is implemented. For such a study, a simplified version of the algorithm was used [22], in which the series is identified with one subsequence, the MPI communication library calls are not involved, and the cycle of processing subsequences of the series continues until the exhaustion of candidate subsequences is exhausted. The size of the segment of the candidate matrix-the number of rows of the candidate matrix processed by one thread within the same computational node of the cluster system as shown in (16) is set to $s=100$.

4.2. The efficiency of the algorithm on one cluster node

The study of the effectiveness of the algorithm on one node of the cluster was presented in Table 1. The time series of the random walk was obtained artificially based on the random walk model [23]. The time series of the EPG was used in experiments in [10]. Electrical penetration graph (EPG) is a set of signals used by entomologists to compare the behavior of the studied insect with the behavior of macrosteles quadrilineatus cicadas, which are carriers of plant diseases and cause more than 2M\$ worth of damage [11].

The experiments examined the performance and scalability of the B_Match Algorithm 1. Performance is understood as the operating time of the algorithm without taking into account the time of loading data into memory and outputting the result. The scalability of a parallel algorithm means its ability to adequately adapt to an increase in parallel computing elements (processes, processors, and threads) and is characterized by speed-up and parallel efficiency [27]. The speed-up and parallel efficiency of a parallel algorithm running on k threads are calculated as $s(k) = t_1/t_k$ and $e(k) = s(k)/k$ respectively, where t_1 and t_k are the operating time of the algorithm on one and k threads.

The above indicators were considered depending on the change in the parameter r (saco-chiba bandwidth), the values of which were taken infractions of the length of the search query n . The runtime of the B_Match algorithm on multicore platforms was compared with the runtime of the UCR-DTW algorithm [9].

The results of experiments to study the performance of the algorithm are presented in Figure 1(a) and (b). It showed that B_Match is up to 5 times faster than the UCR-DTW algorithm. At the same time, the following two parameters affect the performance of the B_Match algorithm on the platforms of the dual-processor Intel Xeon node and the multi-core Intel Xeon system: saco-chiba bandwidth r and search query length n .

For small values of these parameters (approximately $0 < r \leq 0.5n$ and $n < 512$), the B_Match algorithm on the Intel Xeon dual-processor node platform runs somewhat faster or at approximately the same speed as on the Intel Xeon multi-core system platform. At large values of these parameters ($0.5n < r \leq n$ and $n \geq 512$), the algorithm runs faster on the Intel Xeon. This means that the vectorization capabilities incorporated into the design algorithm are best manifested when the total amount of computations increases. Search queries with a length of $n \geq 512$, as well as the value of the $r=1$, are required in practice in many applications that require the highest possible accuracy in determining the similarity of subsequences, for example, in medicine when examining ECG [28], in entomology [29], in astronomy [30].

Table 1. Datasets for experiments on a single cluster node

View	Dataset	$ T /m$ (points)	$ Q /n$
Random Walk	Synthetic	10^6	128
EPG	Real	2.5×10^5	360

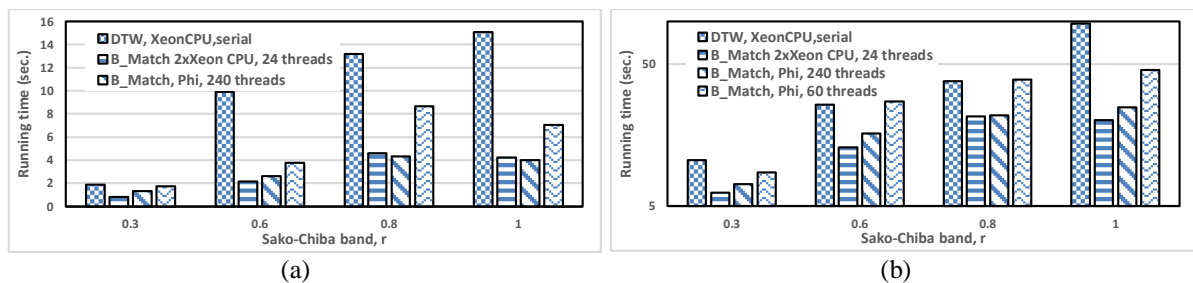


Figure 1. Performance of the B_Match algorithm on a single cluster node with (a) synthetic data set random walk, $m=106$, $n=128$ and (b) a set of real EPG data, $m=2.5 \times 10^5$, $n=360$

The results of experiments to study the scalability of the algorithm are presented in Figures 2 and 3. It can be seen that B_Match shows near-linear speed-up and parallel efficiency close to 100% if the number of threads running the algorithm matches the number of physical cores of the Intel Xeon system. With an increase in the number of threads launched on one physical core of the system, the speed-up becomes sublinear, as well as a decrease in parallel efficiency. At the same time, the best speed-up and parallel efficiency indicators are expected to be observed at values of the parameter $r=0.8$ and 1 of the length of the search query n , which provide the algorithm with the greatest computational load. For example, when 240 threads are used at $r=0.8n$, the processing of the random walk series is performed with a speed-up of 120 and a parallel efficiency of 50%; at $r=n$, the processing of the EPG series is with a speed-up of 130 and a parallel efficiency of 52%.

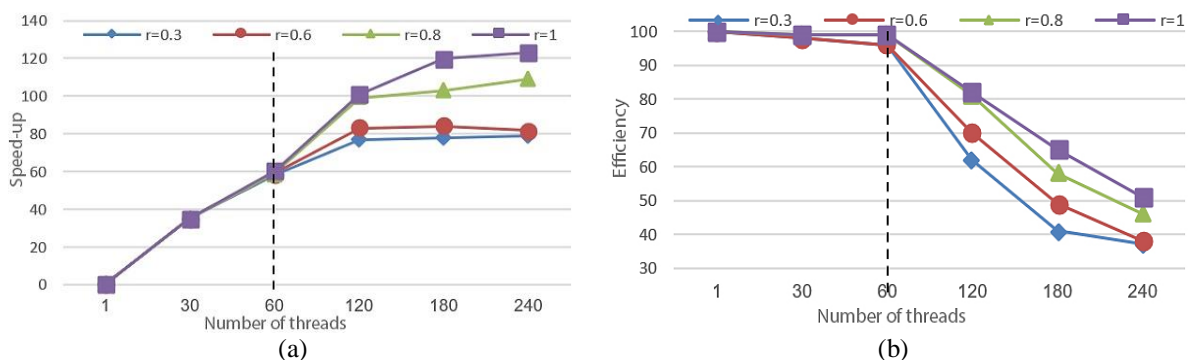


Figure 2. Scalability of the B_Match algorithm on a single cluster node when processing synthetic data (random walk series, $m=106$, $n=128$) by (a) speed-up and (b) parallel efficiency

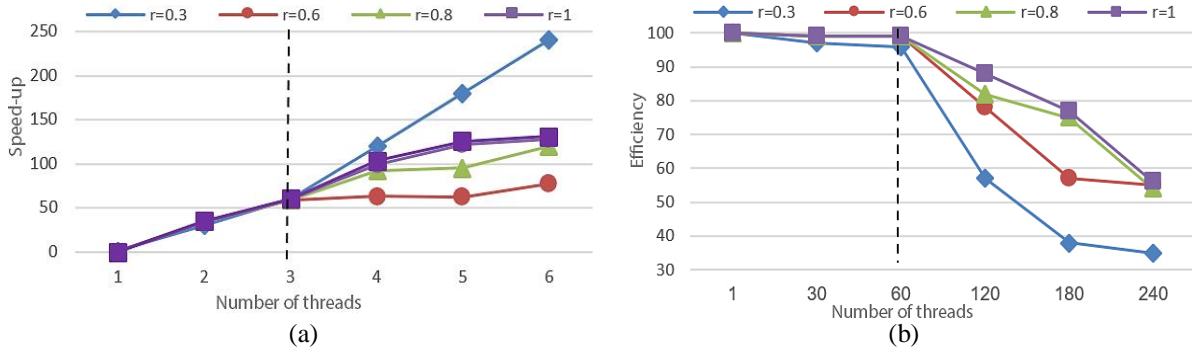


Figure 3. Scalability of the B_Match algorithm on one cluster node when processing real data (EPG series, $m=2.5 \times 10^5$, $n=360$) by (a) speed-up and (b) parallel efficiency

The results obtained allow us to conclude that the developed algorithm has good scalability and its effective use of the capabilities to vectorize computations on the Intel Xeon multicore system within the framework of one computing node of the cluster system. The indicated properties are manifested when the parameters r (saco-chiba bandwidth) and n (search query length) provide the algorithm with the greatest computational load: $0.8n \leq r \leq n$ and $n \geq 512$, respectively.

4.3. The effectiveness of the algorithm on a cluster system

In the study of the effectiveness of the algorithm on a cluster system as a whole, 16-128 nodes of the supercomputer were involved "Tornado SUSU" [30]. Table 2 shows the obtained results for the data sets. In the experiments, we studied the scaled speed-up of the parallel algorithm, which is defined as the speed-up demonstrated by the algorithm with a linear increase in the amount of data and the number of computational nodes used [31] and can be calculated as: $S_{scaled} = p \cdot m / t_{p(p \cdot m)}$ where p is the number of computational nodes involved, m is the volume of the source data, $t_{p(p \cdot m)}$ is the execution time of the algorithm on p nodes when processing the initial data having the volume $p \cdot m$.

Moreover, the value of the parameter r was fixed as $r=n$ and the length of the search query was varied. The results of experiments to study the speed-up of scalability of the algorithm are presented in Figures 4 and 5. It is clear, that B_Match demonstrates near-linear scalability speed-up for both synthetic and real data. At the same time, a search for a subsequence of a greater length shows a higher speed-up of scalability, since this provides a greater amount of computation within a single computational node of a cluster system.

Table 2. Datasets for experiments on a cluster system

Dataset	Type	$ T /m$	$ Q /n$
Random Walk	Synthetic	12.8×10^7	128, 512, 1024
ECG	Real	12.8×10^7	432, 512, 1024

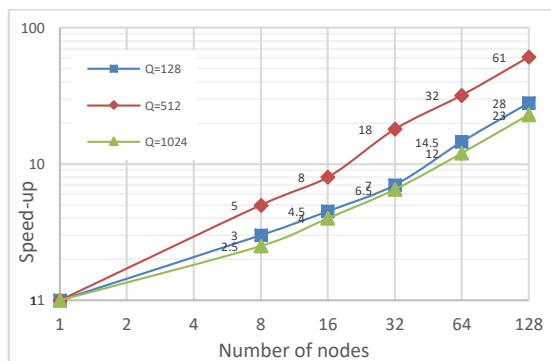


Figure 4. Speed-up of the B_Match algorithm on a cluster system in the processing of synthetic data

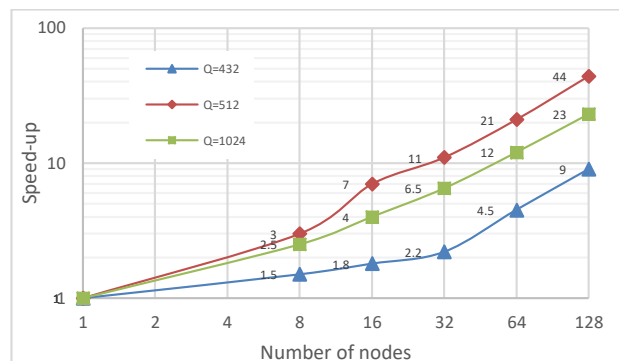


Figure 5. Speed-up of scalability of the B_Match algorithm on a cluster system when processing real data

The results allow us to conclude that the developed algorithm is highly scalable when working on a cluster system with computing nodes based on Intel Xeon multi-core processors. The indicated properties are manifested when the parameters r (SACO-CHIBA bandwidth) and n (search query length) provide the algorithm with the greatest computational load: $0.8n \leq r \leq n$ and $n \geq 512$, respectively.

5. CONCLUSION

The paper considers the problem of searching for similar subsequences in super-large time series (hundreds of billions of points) based on the use of the DTW similarity measure (dynamic transformation of the time scale). This problem arises in a wide range of applications of the intellectual analysis of time series: climate modeling, financial forecasts, medical research. A new parallel algorithm is proposed for searching for similar subsequences of a time series on a cluster system with computing nodes based on Intel Xeon multi-core processors of the knight's landing generation, called B_Match. The algorithm assumes a two-level parallelization of computations: MPI technology is used at the level of all nodes of the cluster system, and OpenMP technology is used within the same computing node of the cluster. To effectively use the capabilities of vectorization of computations of KNL processors within the framework of a single computing node, additional matrix data structures, and redundant calculations are used. Computational experiments were carried out to study the speed and scalability of the algorithm on synthetic and real data sets both on the cluster system as a whole and within the framework of one computing node of the cluster. The experimental results showed good scalability of the B_Match algorithm on one node and on the cluster system as a whole.

ACKNOWLEDGMENTS

The authors would like to thank Mosul University, Iraq and Bibliotheca Alexandria, Egypt for granting the access for running their computations on its platform.




REFERENCES

- [1] B. Peachey Higdon, K. El Mokhtari, and A. Başar, "Time-Series-Based Classification of Financial Forecasting Discrepancies," *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 2019, pp. 474–479, doi: 10.1007/978-3-030-34885-4_39.
- [2] H. Rathore, A. Al-Ali, A. Mohamed, X. Du, and M. Guizani, "DTW based Authentication for Wireless Medical Device Security," in *2018 14th Int. Wir.Comm. Mob. Comp. Conf. (IWCMC)*, Jun. 2018, pp. 476–481, doi: 10.1109/IWCMC.2018.8450419.
- [3] I. Bjelanovic, P. Comeau, and B. White, "High Resolution Site Index Prediction in Boreal Forests Using Topographic and Wet Areas Mapping Attributes," *Forests*, vol. 9, no. 3, p. 113, Mar. 2018, doi: 10.3390/f9030113.
- [4] F. Albu, D. Hagiescu, M. Puica, and L. Vladutu, "Intelligent tutor for first grade children's handwriting application," *Conference: INTED 2015: 9th International Technology, Education and Development Conference Madrid, Spain*, 2015.
- [5] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," *KDD workshop*, vol. 10, no. 16, pp. 359–370, 1994.
- [6] J. Rodrigues, D. Folgado, D. Belo, and H. Gamboa, "SSTS: A syntactic tool for pattern search on time series," *Information Processing & Management*, vol. 56, no. 1, pp. 61–76, Jan. 2019, doi: 10.1016/j.ipm.2018.09.001.
- [7] C. W. Tan, G. I. Webb, and F. Petitjean, "Indexing and classifying gigabytes of time series under time warping," in *Proc. 2017 SIAM Int. Conference on Data Mining*, Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017, pp. 282–290.
- [8] K. Stadler et al., "EXIOBASE 3: Developing a Time Series of Detailed Environmentally Extended Multi-Regional Input-Output Tables," *Journal of Industrial Ecology*, vol. 22, no. 3, pp. 502–515, Jun. 2018, doi: 10.1111/jiec.12715.
- [9] D. F. Silva, R. Giusti, E. Keogh, and G. E. A. P. A. Batista, "Speeding up similarity search under dynamic time warping by pruning unpromising alignments," *Data Min. Know. Disc.*, vol. 32, no. 4, pp. 988–1016, 2018, doi: 10.1007/s10618-018-0557-y.
- [10] X. Xu et al., "Accelerating Dynamic Time Warping With Memristor-Based Customized Fabrics," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 4, pp. 729–741, Apr. 2018, doi: 10.1109/TCAD.2017.2729344.
- [11] M. W. Al-Neama, N. M. Reda, and F. F. M. Ghaleb, "Fast vectorized distance matrix computation for multiple sequence alignment on multi-cores," *Int. J. Biomathematics*, vol. 08, no. 06, p. 1550084, Nov. 2015, doi: 10.1142/S1793524515500849.
- [12] S. Kang, J. Moon, and S.-W. Jun, "FPGA-Accelerated Time Series Mining on Low-Power IoT Devices," in *2020 IEEE 31st Int. Conf. App.-specific Sys., Architectures and Processors (ASAP)*, Jul. 2020, pp. 33–36, doi: 10.1109/ASAP49362.2020.00015.
- [13] V. S. Siyow Fotso, E. Mephu Nguifo, and P. Vasilin, "Frobenius correlation based u-shapelets discovery for time series clustering," *Pattern Recognition*, vol. 103, p. 107301, Jul. 2020, doi: 10.1016/j.patcog.2020.107301.
- [14] M. Govett et al., "Parallelization and Performance of the NIM Weather Model on CPU, GPU, and MIC Processors," *Bulletin of the American Meteorological Society*, vol. 98, no. 10, pp. 2201–2213, Oct. 2017, doi: 10.1175/BAMS-D-15-00278.1.
- [15] B. Plazolles, D. El Baz, M. Spel, V. Rivola, and P. Gegout, "SIMD Monte-Carlo Numerical Simulations Accelerated on GPU and Xeon Phi," *Int. Journal of Parallel Programming*, vol. 46, no. 3, pp. 584–606, Jun. 2018, doi: 10.1007/s10766-017-0509-y.
- [16] S. Ramos and T. Hoefler, "Capability Models for Manycore Memory Systems: A Case-Study with Xeon Phi KNL," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2017, pp. 297–306, doi: 10.1109/IPDPS.2017.30.
- [17] I. Sokolinskaya and L. B. Sokolinsky, "Scalability Evaluation of NSLP Algorithm for Solving Non-Stationary Linear Programming Problems on Cluster Computing Systems," 2017, pp. 40–53.
- [18] D. F. Bacon, S. L. Graham, and O. J. Sharp, "Compiler transformations for high-performance computing," *ACM Computing Surveys*, vol. 26, no. 4, pp. 345–420, Dec. 1994, doi: 10.1145/197405.197406.
- [19] Y. Kraeva and M. Zymbler, "Scalable Algorithm for Subsequence Similarity Search in Very Large Time Series Data on Cluster of Phi KNL," 2019, pp. 149–164.




- [20] A. Shabib *et al.*, "Parallelization of searching and mining time series data using Dynamic Time Warping," in *2015 Int. Conf. Adv. Computing, Communications and Informatics (ICACCI)*, Aug. 2015, pp. 343–348, doi: 10.1109/ICACCI.2015.7275633.
- [21] M. Zymbler, "Best-Match Time Series Subsequence Search on the Intel Many Integrated Core Architecture," 2015, pp. 275–286.
- [22] Y. Kraeva and M. L. Zymbler, "An Efficient Subsequence Similarity Search on Modern Intel Many-core Processors for Data Intensive Applications," 2018.
- [23] Y. Hwang and S. B. Gelfand, "Constrained Sparse Dynamic Time Warping," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2018, pp. 216–222, doi: 10.1109/ICMLA.2018.00039.
- [24] J. Tarango, E. Keogh, and P. Brisk, "Instruction set extensions for Dynamic Time Warping," in *2013 Int. Conf. Hardware/Software Codesign and Sys. Synt. (CODES+ISSS)*, Sep. 2013, pp. 1–10, doi: 10.1109/CODES-ISSS.2013.6659005.
- [25] W. Gropp, "MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces," *EuroMPI 2012: Recent Advances in the Message Passing Interface*, 2012, pp. 1–9, doi: 10.1007/978-3-642-33518-1_1.
- [26] Я. А. Краева and М. Л. Цымблер, "The use of MPI and OpenMP technologies for subsequence similarity search in very long time series on a computer cluster system with nodes based on the Intel Xeon Phi Knights Landing many-core processor," *Num. Meth. Progr. (Vychislitel'nye Metody i Programirovanie)*, no. 1, pp. 29–44, Jan. 2019, doi: 10.26089/NumMet.v20r104.
- [27] "Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community," *Supercomputing Frontiers and Innovations*, vol. 6, no. 2, Jun. 2019, doi: 10.14529/jsfi190201.
- [28] F. N. Mazandarani and M. Mohebbi, "Wide complex tachycardia discrimination using dynamic time warping of ECG beats," *Computer Methods and Programs in Biomedicine*, vol. 164, pp. 238–249, Oct. 2018, doi: 10.1016/j.cmpb.2018.04.009.
- [29] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos, "Approximate embedding-based subsequence matching of time series," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, 2008, p. 365, doi: 10.1145/1376616.1376656.
- [30] P. Kostenetskiy and P. Semenikhina, "SUSU Supercomputer Resources for Industry and fundamental Science," in *2018 Global Smart Industry Conference (GloSIC)*, Nov. 2018, pp. 1–7, doi: 10.1109/GloSIC.2018.8570068.
- [31] A. M. Alatwi, A. N. Zaki Rashed, A. M. El-Eraki, and I. Sadeh Amiri, "Best candidate routing algorithms integrated with minimum processing time and low blocking probability for modern parallel computing systems," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 2, p. 847, Aug. 2020, doi: 10.11591/ijeecs.v19.i2.pp847-854.

BIOGRAPHIES OF AUTHORS






Firas Mahmood Saeed    received the MSc degree in computing mathematics from Mosul University. He currently works at the Medicine College, University of Mosul. He can be contacted at email: fms@uomosul.edu.iq.



Salwa M. Ali    has obtained the Ph.D in computer science from Ain Shams University, Cairo (Egypt). She is currently worked at Unaizah College of Sciences and Arts, Qassim University, (K.S.A.). Her research area includes information science foundation, lambda calculus. She can be contacted at email: s.mussa@qu.edu.sa.



Mohammed Wajid Al-Neama    has a Ph.D in computing mathematics from Al-Azhar University, Cairo (Egypt) in 2014. He currently works at the Education College for Girls, Mosul University. His research area includes Bioinformatics, Parallel Computing. He can be contacted at email: mwneama@uomosul.edu.iq.